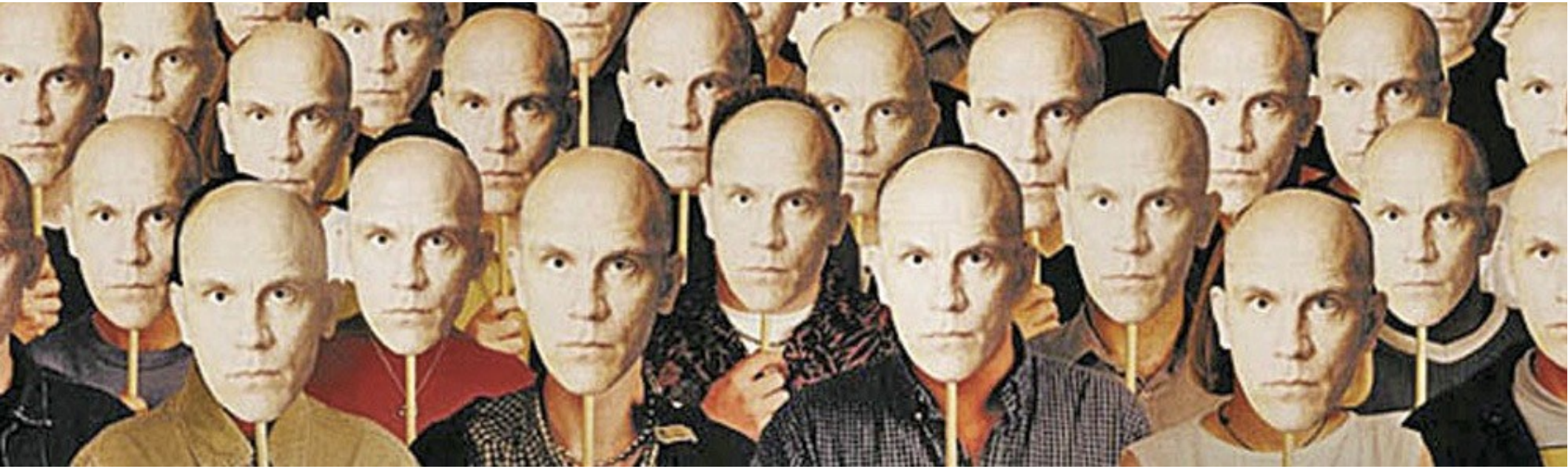
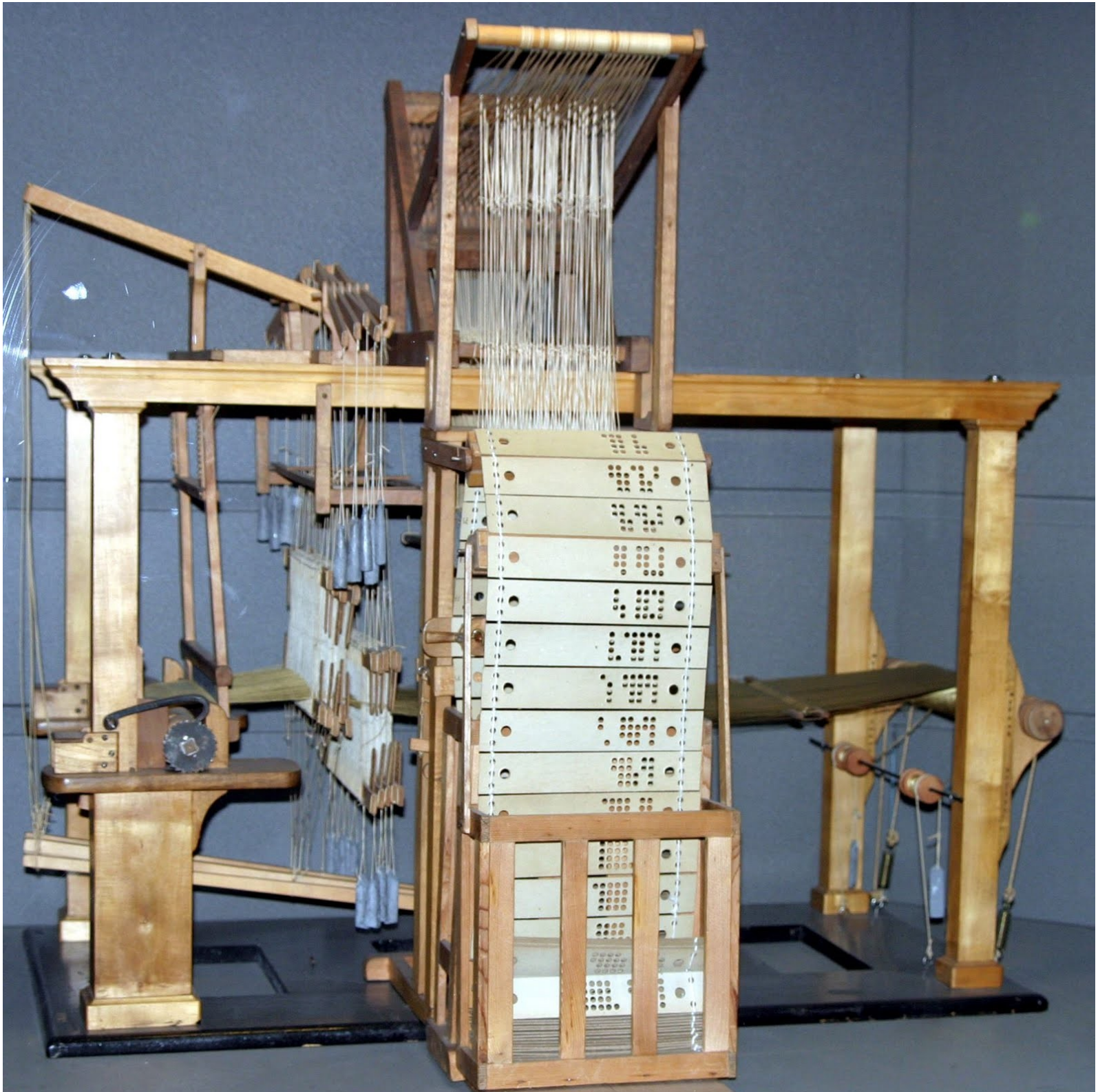
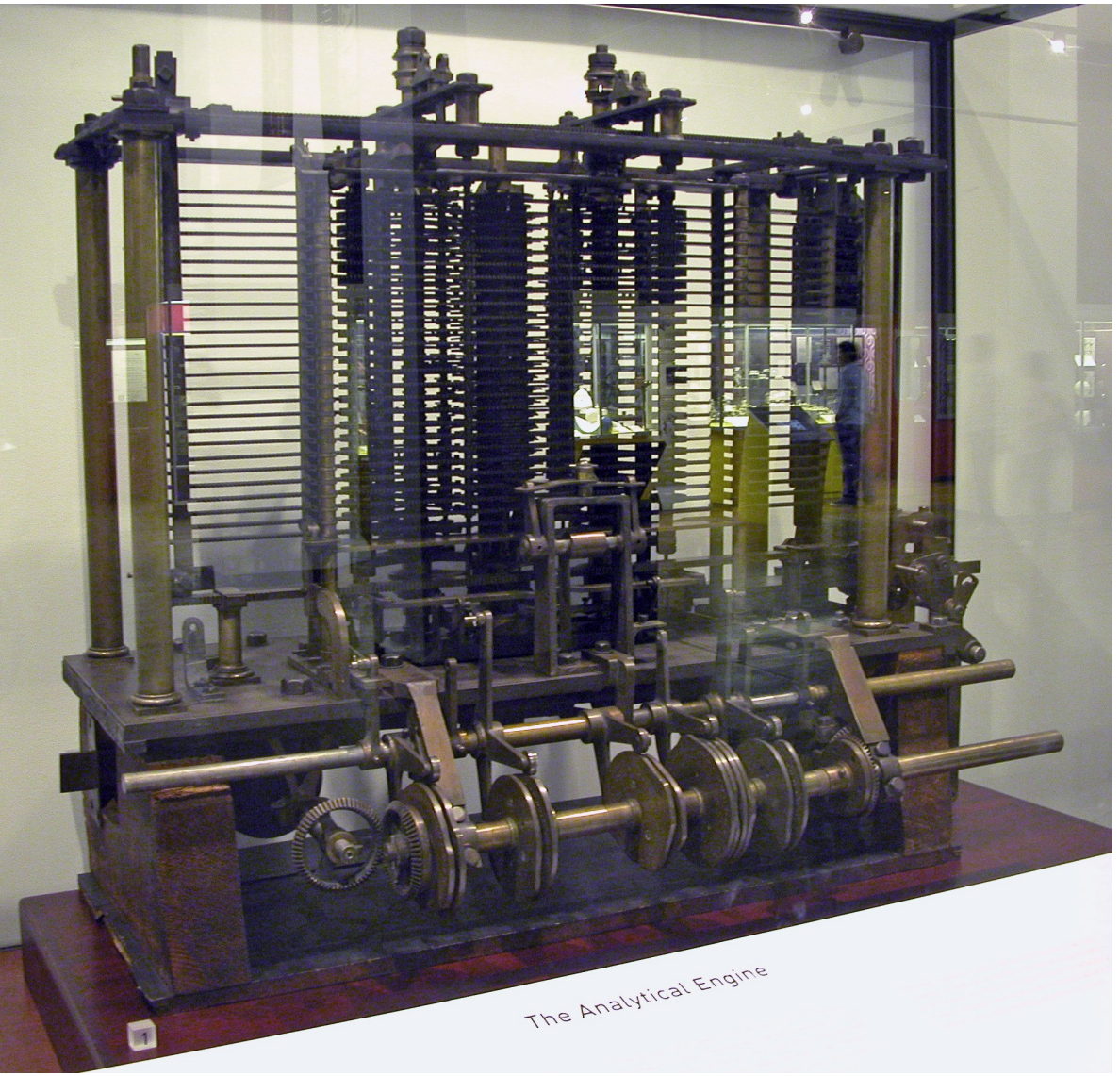
Repeat! Repeat! Repeat!





Jacquard loom



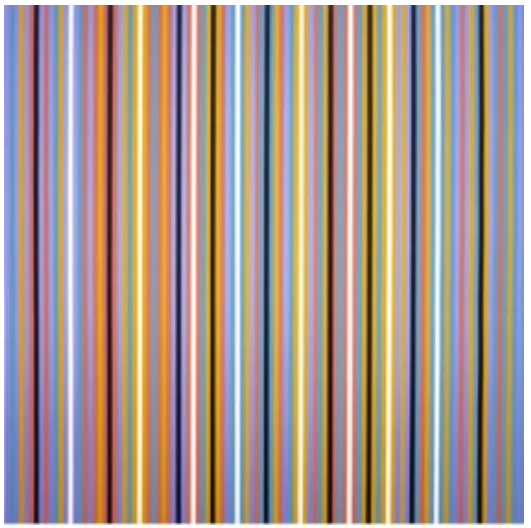
Charles Babbage's analytical engine

- computers are exceptional for creating repetition
- repetition is still an inherent part of code – and a great inspiration

Repetition

- can be incredibly powerful
  - flashing light can induce seizures
  - repetition important in music - shape of a song
  - a good beat can inspire dance
  - dynamic patterns can appear to vibrate
- optical art – 60's movement

Bridget Riley





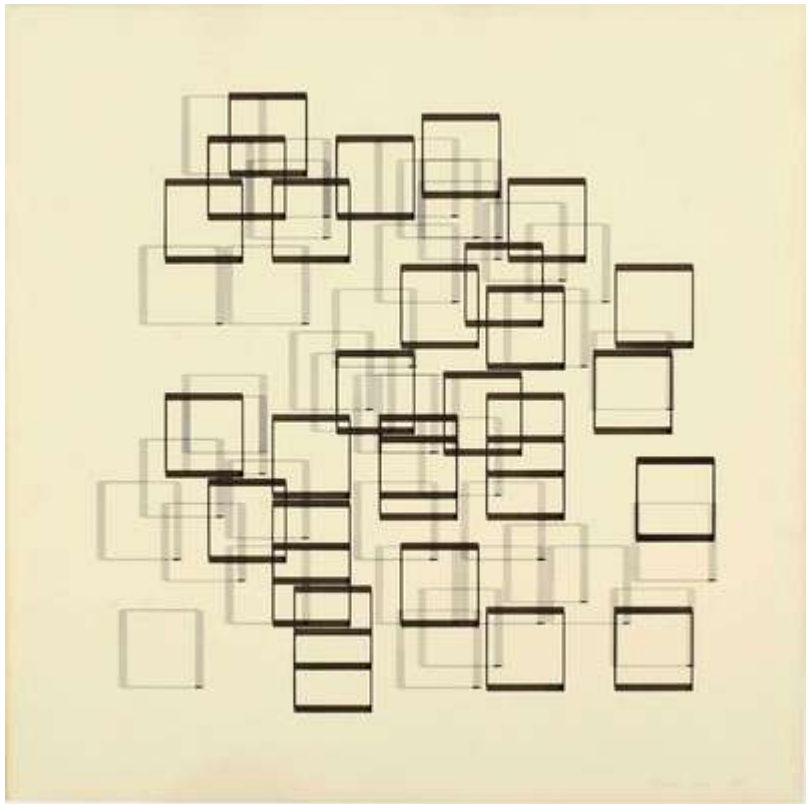


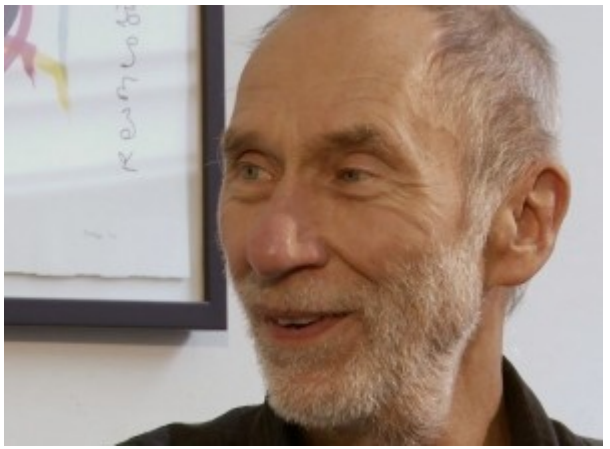


Campbell's
CONDENSED
ONION
MADE WITH BEEF STOCK
SOUP
Campbell's
CONDENSED
CREAM OF
MUSHROOM
SOUP
Campbell's
CONDENSED
BEEF
WITH VEGETABLES AND BARLEY
SOUP
Campbell's
CONDENSED
CONSOMMÉ
(BEEF)
GELATIN ADDED
SOUP
Campbell's
CONDENSED
GREEN PEA
SOUP
Campbell's
CONDENSED
PEPPER POT
SOUP
Campbell's
CONDENSED
VEGETABLE
MADE WITH BEEF STOCK
SOUP
Campbell's
CONDENSED
CHICKEN
NOODLE
SOUP
Campbell's
CONDENSED
TOMATO
SOUP
Campbell's
CONDENSED
BLACK BEAN
SOUP

Early computer reproductions

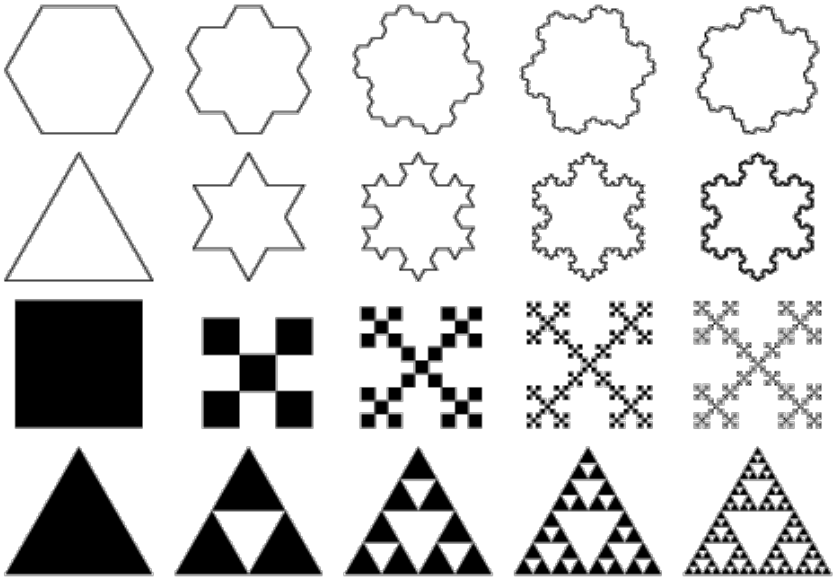
- computers are designed to accurately perform the same calculation.
- difficult to work against machine's electronic precision to produce idiosyncratic images
- Frieder Nake (U of Stuttgart) among the first to use a pen plotter to produce drawings from source for aesthetic reasons



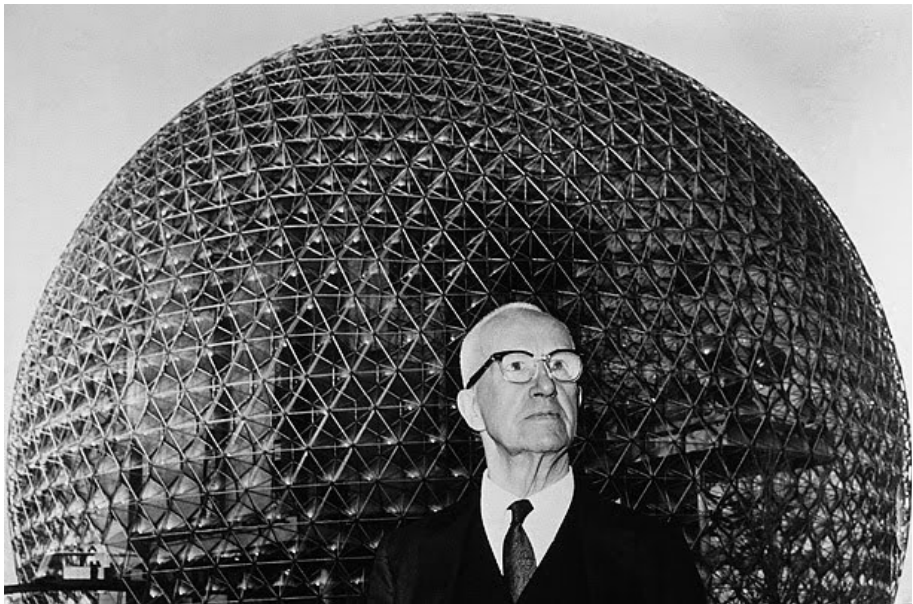


Modularity

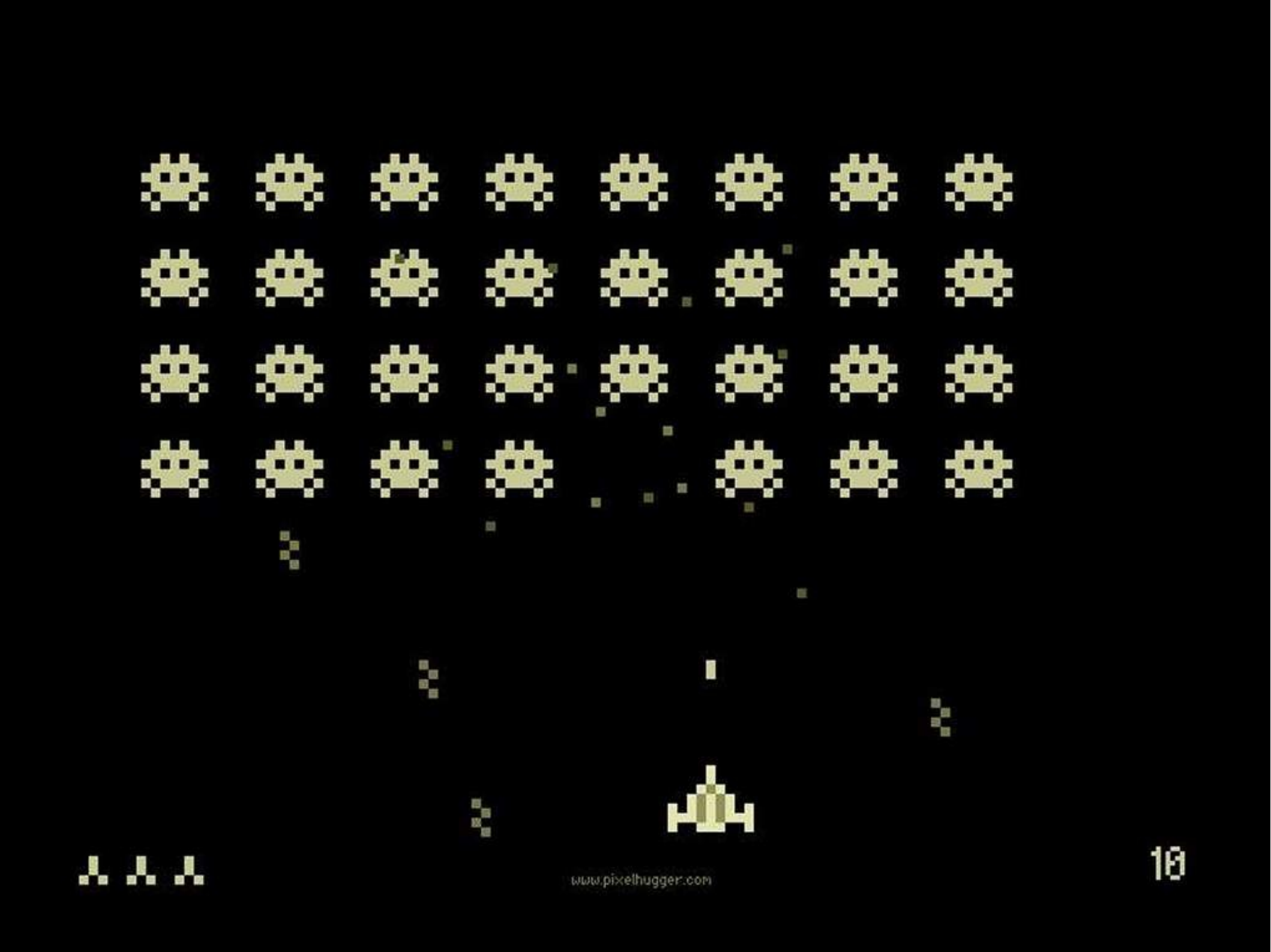
- in typography
  - typefaces are very modular
- in software it is used for optimization
  - produce complex images from small group of forms
    - 90's site wallpapers
    - fractals
- in construction
  - buildings are made by standardized elements
  - Buckminster Fuller took the idea to the extreme



abcdefghijklm
nopqrstuvwxyz

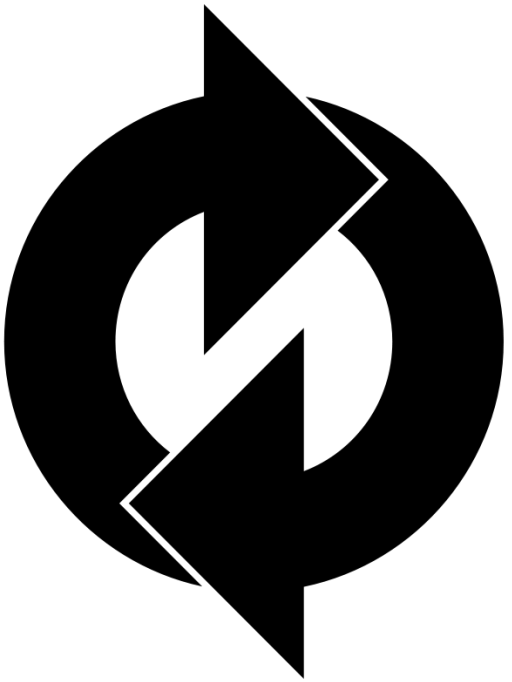



introduction
to loops


www.pixelhugger.com
10

Loops

- we use loops since we learned about animation
- `draw()` is a loop, but:
  - it never stops
  - there is only one `draw()` in each sketch
  - it changes elements on screen
  - it's slow (<=60 fps)



while

"While it's raining I am going to stay home"

```
while ( raining )    // evaluates statement
{
   - stay home
 }
```

while()

```
void ofApp::draw()
{
  int circleX = 10;

  while (circleX < ofGetWidth())
  {
    ofCircle(circleX, ofGetHeight()/2, 20, 10);
    circleX = circleX + 50;
  }
}
```

initialization

evaluating the conditional

updating the variable

Loops - while()

```
void ofApp::draw()
{
  int circleX = 10;

  while (circleX < ofGetWidth())
  {
    ofCircle(circleX, ofGetHeight()/2, 10);
    circleX = circleX + 50;
  }
}
```

initialization

evaluating the conditional

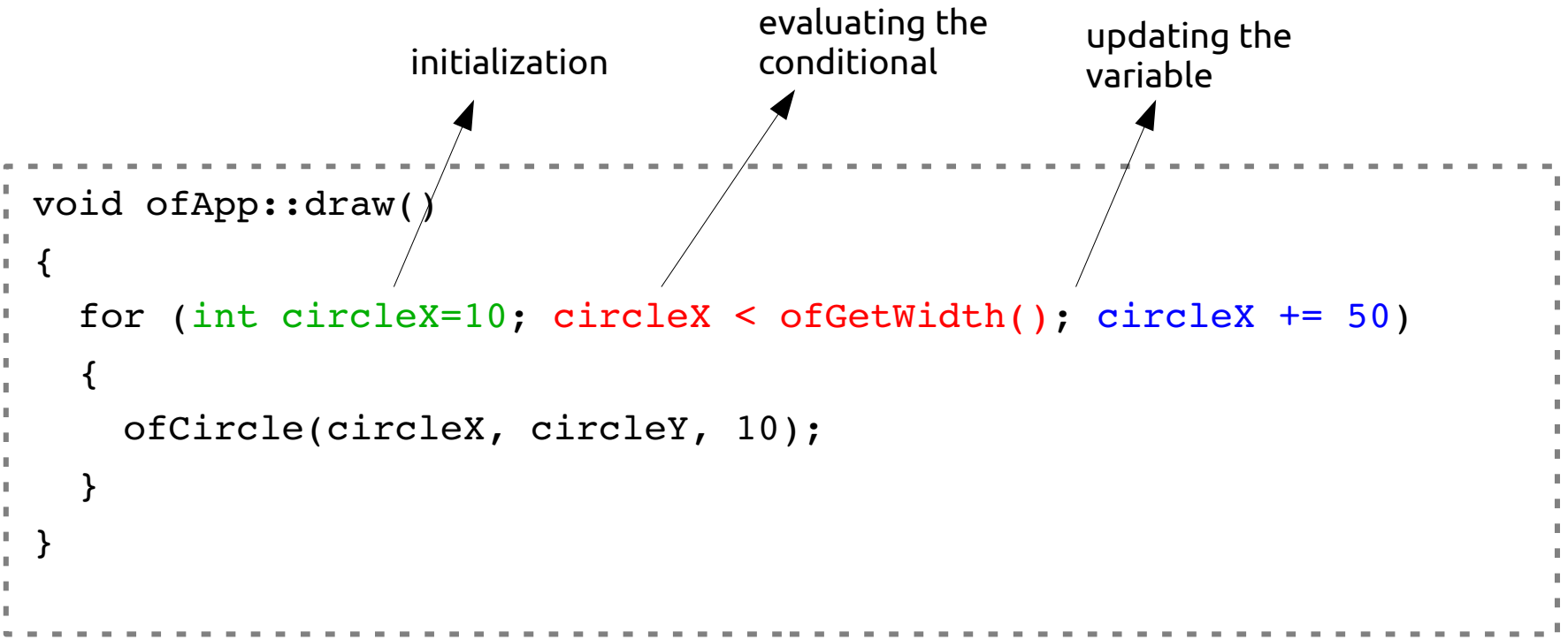
updating the variable

Loops – `while`

if the conditional is always true, the program is going to run forever


BLOODY MARY,
BLOODY MARY,
BLOODY MARY.
RAAGHHHHH

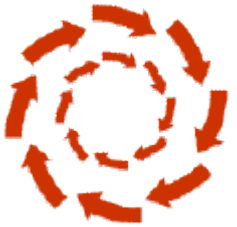
Loops - for()


initialization
evaluating the conditional
updating the variable
void ofApp::draw()
{
for (int circleX=10; circleX < ofGetWidth(); circleX += 50)
{
ofCircle(circleX, circleY, 10);
}
}

Loops - for()

```
void ofApp::draw()
{
  for (int circleX=10; circleX < width; circleX = circleX + 30)
  {
    ellipse(circleX, circleY, 20, 20);
  }
}
```

Loops - for



- updating the variable happens at the end of each cycle
- the variable that is initialized is local (it's known only within the loop, we can't use it outside)
- we can build a loop within loop if we want to add extra dimensions

```
for (int x = 0; x < 10; x++)
{
  for (int y = 0; y < 10; y++)
  {
     // code to execute
  }
}
```

scope

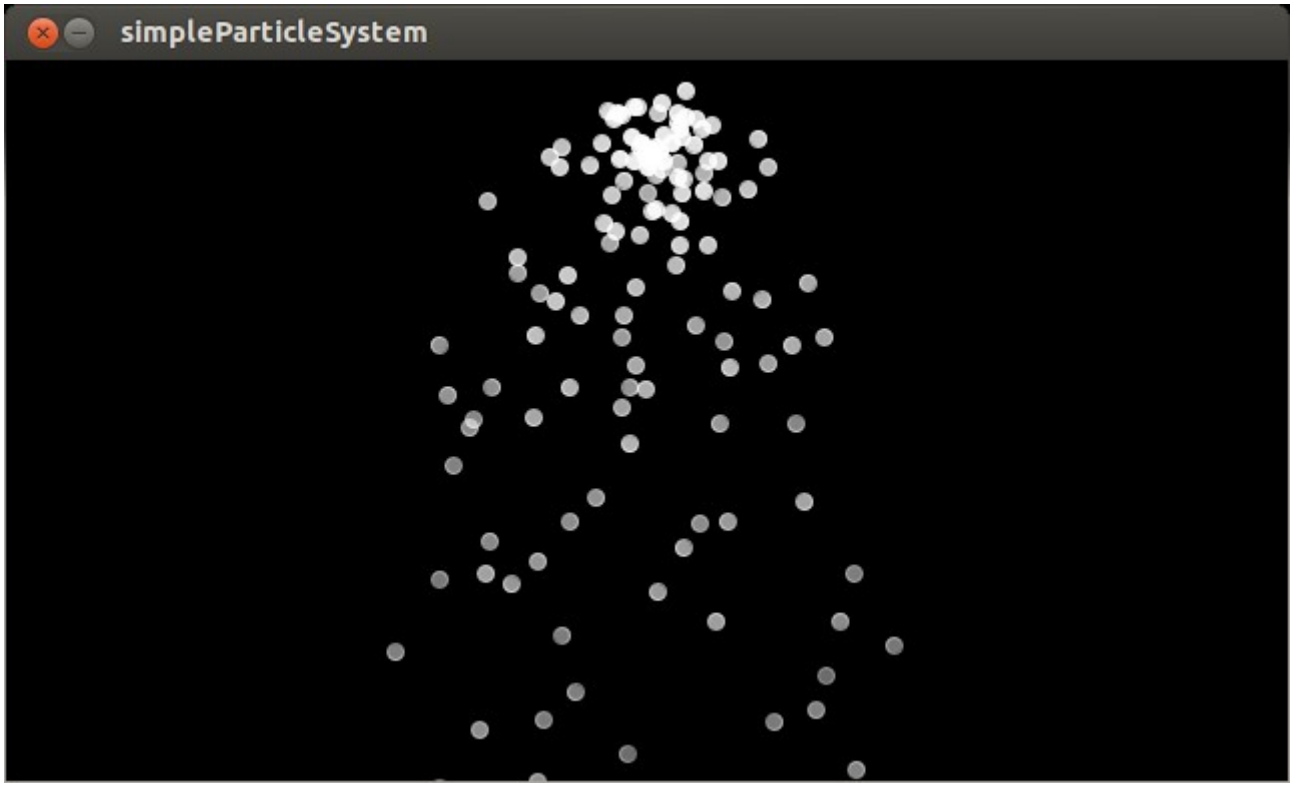
```
int x = 2;

void ofApp::setup()
{
    int x = 3;
    println(x);                 // ???
}

void ofApp::draw()
{
     println(x);                    // ???
}
```

scope (cont.)

```
int x = 2;

void ofApp::setup()
{
    int x = 3;
    cout << x;                      // ???
}

void ofApp::draw()
{
    for (int x = 5; x < y; x++)
    {
        cout << x;                  // ???
    }
}
```

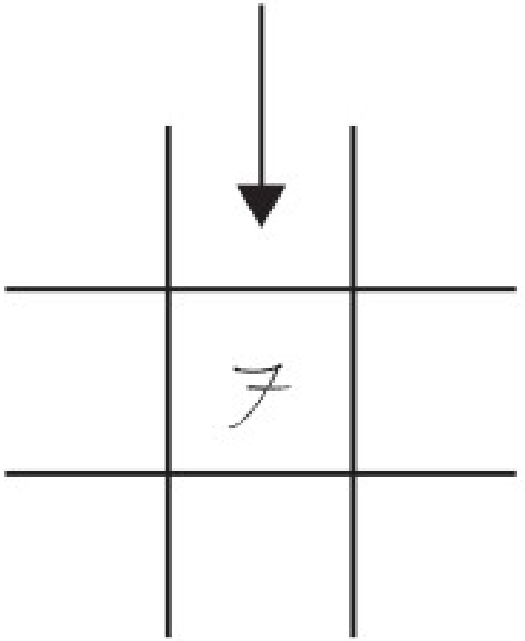
vectors

“Should array indices start at 0 or 1? My compromise of 0.5 was rejected without, I thought, proper consideration.”
Stan Kelly-Bootle



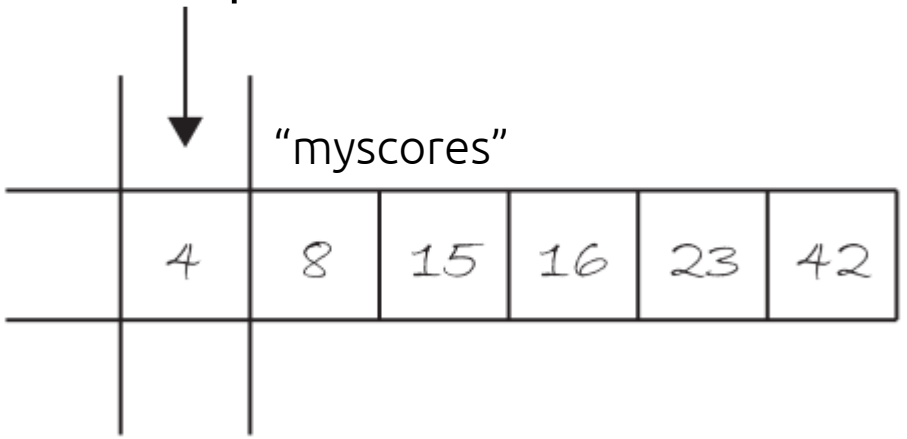
examples/math/particlesExample

```
int myscore;
```

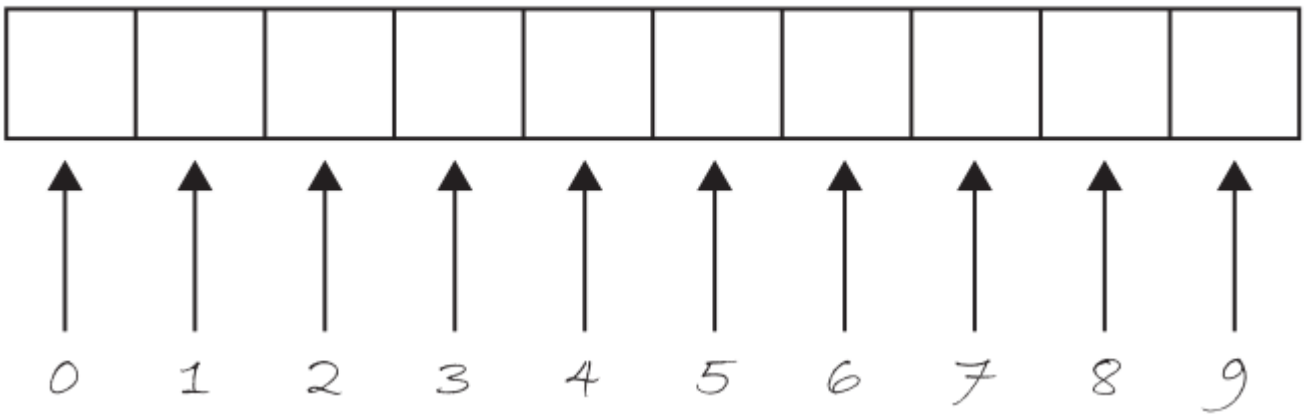


vector

value at position 0



index of values



How do we declare an array;

```
vector<int> myGrades;                  // vector declaration
myScores.push_back(80);                // adding value at the end of vector
myScores[0];                           // read the first value
cout << myScores.size() << endl;       // getting vector size
myScores.pop_back();                   // popping last value
```

...other examples

```
vector<int> myGrades;
vector<float> myTemperatures;
vector<ofColor> myColors;
vector<ofRectangle> myRectangles;
```

Question: What do we do if we have a vector with 1000 elements?

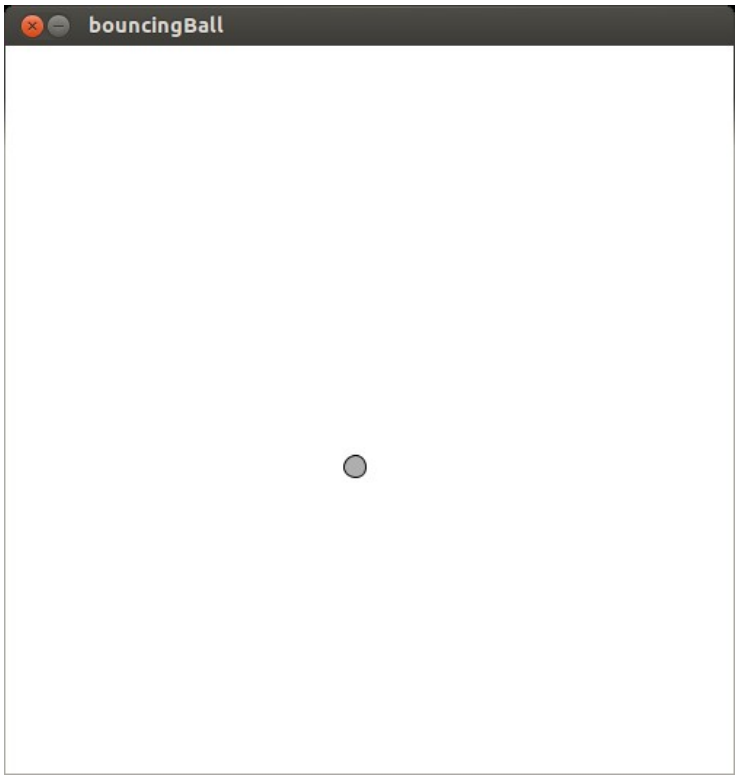
```
vector <float> circleX;

circleX.push_back(ofRandom(0,200));

circleX.push_back(ofRandom(0,200));

circleX.push_back(ofRandom(0,200));

…

circleX.push_back(ofRandom(0,200));
```

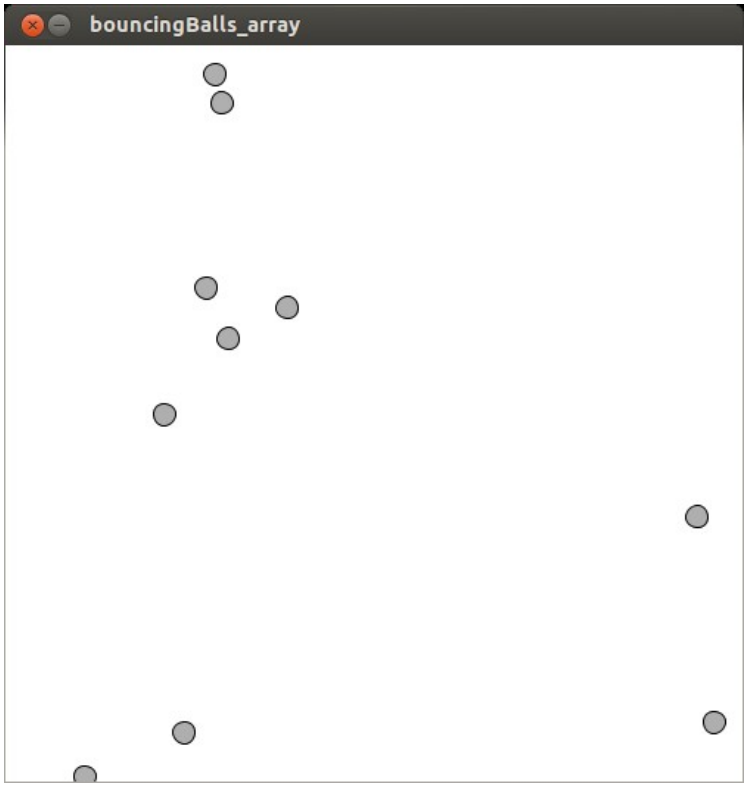
Answer

We can use a for loop!!!!

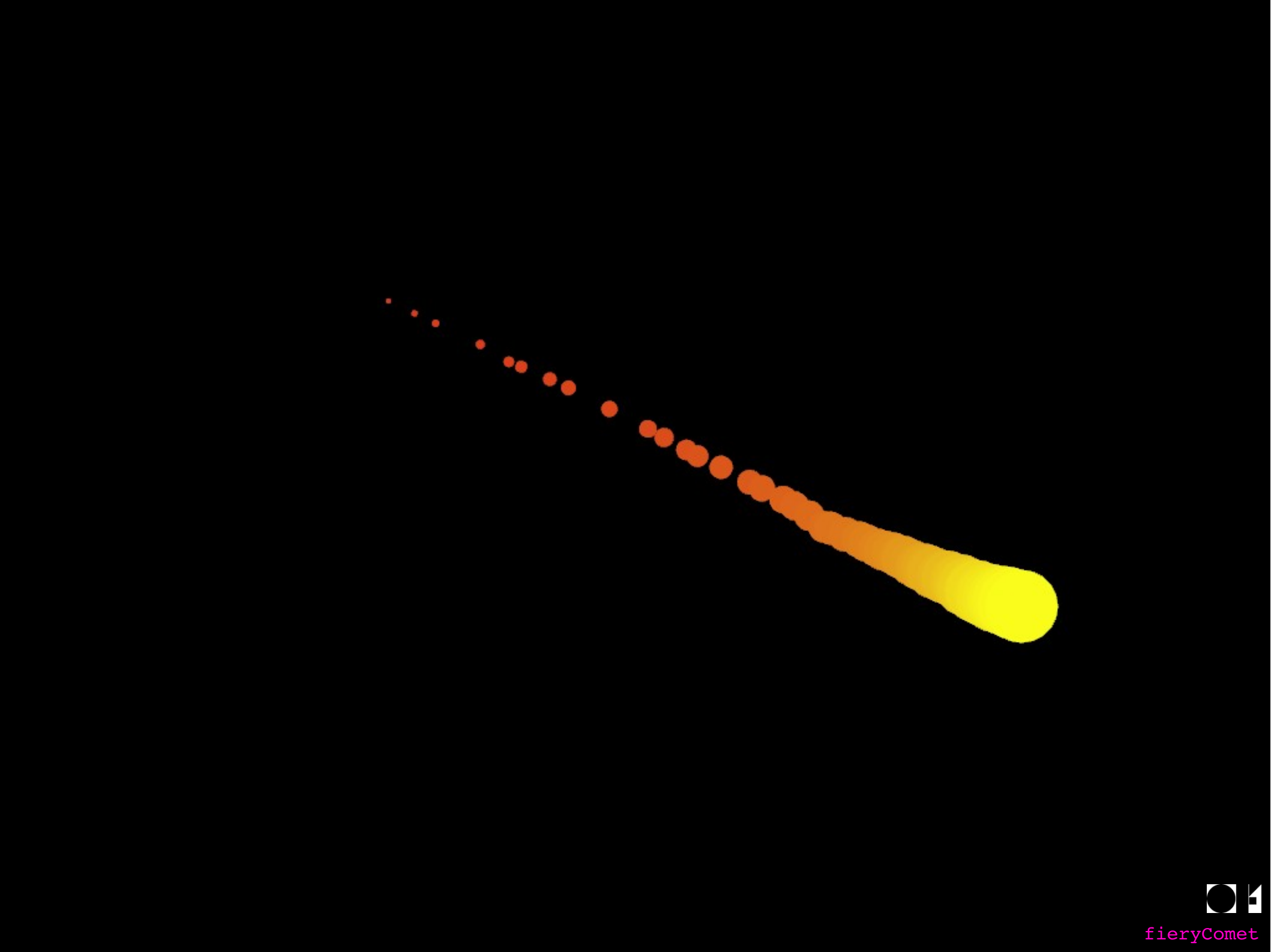
```
vector<float> circleX;

for (int i = 0; i<10000; i++)
{
     circleX[i] = ofRandom(0,200);
}
```

bouncing balls



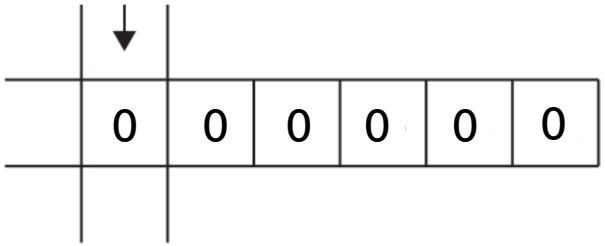


bouncingBall
bouncingBallVector

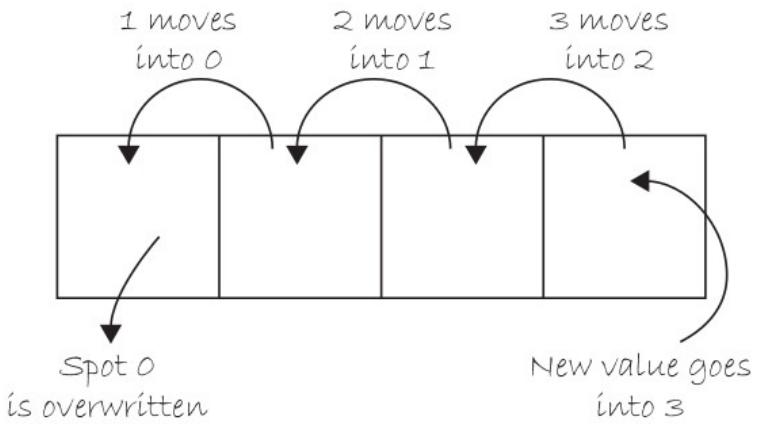

fieryComet

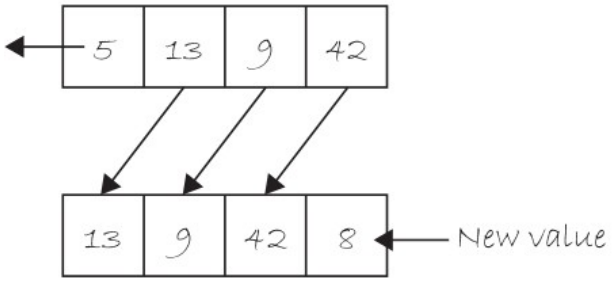
fiery comet
behind the scenes

1. Create an array and I initialize the values



2. In each frame of draw() we need to update the array with the latest position of the mouse





3. Draw ofCircles() where the points are

vector operations

- size( )
- push_back()
- pop_back()
- erase(.....)
- other operations

a new transformation

- `ofScale(float amtX, float amtY)`
  - the order **does** matter (again!)

building oFx examples